Exercise 1: Setting Up Junit

Pom.xml :

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>calculator-app</artifactId>

<version>1.0-SNAPSHOT</version>

<name>CalculatorApp</name>

<dependencies>

<!-- JUnit 5 Dependency -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

src/main/java/com/example/Calculator.java:

java

CopyEdit

package com.example;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

### src/test/java/com/example/CalculatorTest.java :

java

CopyEdit

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

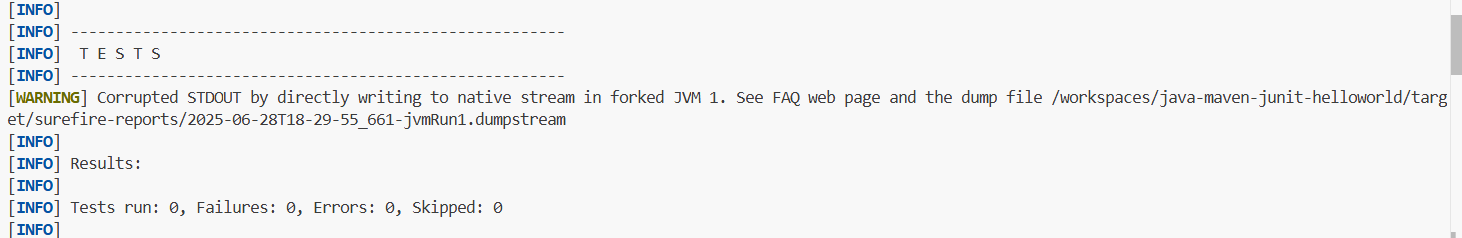
int result = calc.add(2, 3);

assertEquals(5, result);

}

}

Output:



Exercise 3: Assertions in Junit

## AssertionsTest.java:

java

CopyEdit

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

// Assert equals

assertEquals(5, 2 + 3);

// Assert true

assertTrue(5 > 3);

// Assert false

assertFalse(5 < 3);

// Assert null

assertNull(null);

// Assert not null

assertNotNull(new Object());

}

}

## pom.xml :

xml

CopyEdit

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>junit-assertions-example</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.9.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Surefire Plugin to run JUnit 5 tests -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

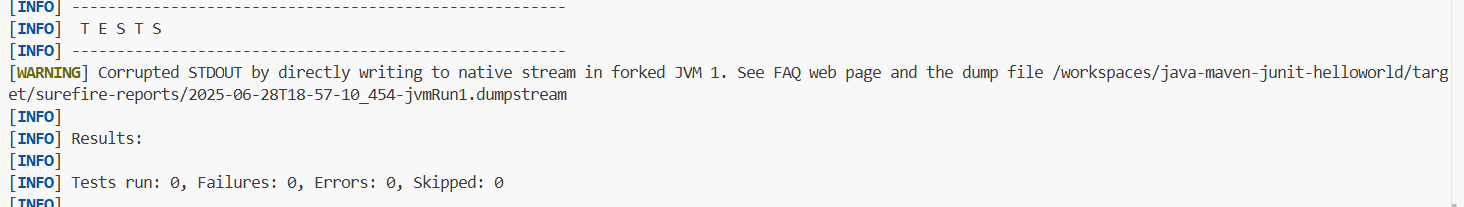
</plugin>

</plugins>

</build>

</project>

Output:



Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

Calculator.java:

package com.example.calculatorapp;

public class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

    public int subtract(int a, int b) {

        return a - b;

    }

}

Calculatortest.java:

package com.example.calculatorapp;

import org.junit.jupiter.api.\*;

import static org.junit.jupiter.api.Assertions.\*;

public class CalculatorTest {

    private Calculator calculator;

    @BeforeEach

    public void setUp() {

        calculator = new Calculator();

        System.out.println("Setup complete");

    }

    @AfterEach

    public void tearDown() {

        calculator = null;

        System.out.println("Teardown complete");

    }

    @Test

    public void testAddition() {

        int result = calculator.add(2, 3);

        assertEquals(5, result);

    }

    @Test

    public void testSubtraction() {

        int result = calculator.subtract(5, 3);

        assertEquals(2, result);

    }

}

Pom.xml:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>java-maven-junit-helloworld</artifactId>

  <version>2.0-SNAPSHOT</version>

  <packaging>jar</packaging>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

    <!-- This configures the compiler to produce Java 8 class files. -->

    <!-- The minimum JDK version installed is 8 of course, but newer JDK releases should work too. -->

    <maven.compiler.source>1.8</maven.compiler.source>

    <maven.compiler.target>${maven.compiler.source}</maven.compiler.target>

    <junit.jupiter.version>5.2.0</junit.jupiter.version>

    <junit.platform.version>1.2.0</junit.platform.version>

    <hamcrest.version>1.3</hamcrest.version>

    <mockito.version>2.21.0</mockito.version>

    <jacoco.plugin.version>0.8.1</jacoco.plugin.version>

  </properties>

  <dependencies>

    <!-- Testing dependencies. -->

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter-api</artifactId>

      <version>${junit.jupiter.version}</version>

      <scope>test</scope>

    </dependency>

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter-params</artifactId>

      <version>${junit.jupiter.version}</version>

      <scope>test</scope>

    </dependency>

    <dependency>

      <groupId>org.junit.jupiter</groupId>

      <artifactId>junit-jupiter-engine</artifactId>

      <version>${junit.jupiter.version}</version>

      <scope>test</scope>

    </dependency>

    <dependency>

      <groupId>org.hamcrest</groupId>

      <artifactId>hamcrest-core</artifactId>

      <version>${hamcrest.version}</version>

    </dependency>

    <dependency>

      <groupId>org.mockito</groupId>

      <artifactId>mockito-junit-jupiter</artifactId>

      <version>${mockito.version}</version>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <plugin>

        <!-- Configures the compiler. -->

        <groupId>org.apache.maven.plugins</groupId>

        <artifactId>maven-compiler-plugin</artifactId>

        <version>3.8.0</version>

        <configuration>

          <compilerArgs>

            <arg>-Xlint</arg>

          </compilerArgs>

        </configuration>

      </plugin>

      <plugin>

        <!-- Unit tests are run by surefire. -->

        <!-- Classes under src/test/java called \*Test are included automatically. -->

        <!-- Unit tests are run during the test phase. -->

        <groupId>org.apache.maven.plugins</groupId>

        <artifactId>maven-surefire-plugin</artifactId>

        <version>2.22.0</version>

      </plugin>

      <plugin>

        <!-- Integration tests are run by failsafe. -->

        <!-- Classes under src/test/java called \*IT are included automatically. -->

        <!-- Integration tests are run during the verify phase. -->

        <groupId>org.apache.maven.plugins</groupId>

        <artifactId>maven-failsafe-plugin</artifactId>

        <version>2.22.0</version>

        <executions>

          <execution>

            <goals>

              <goal>integration-test</goal>

              <goal>verify</goal>

            </goals>

          </execution>

        </executions>

      </plugin>

      <plugin>

        <!-- Code coverage plugin. -->

        <!-- This sets up code coverage for the unit tests, the integration tests, -->

        <!-- and the combined coverage of both. -->

        <!-- After running a `mvn clean install`, open `target/site/jacoco-both/index.html` in a browser. -->

        <groupId>org.jacoco</groupId>

        <artifactId>jacoco-maven-plugin</artifactId>

        <version>${jacoco.plugin.version}</version>

        <executions>

          <execution>

            <id>jacoco-prepare-agent</id>

            <goals>

              <goal>prepare-agent</goal>

            </goals>

          </execution>

          <execution>

            <id>jacoco-prepare-agent-integration</id>

            <goals>

              <goal>prepare-agent-integration</goal>

            </goals>

          </execution>

          <execution>

            <id>jacoco-report</id>

            <goals>

              <goal>report</goal>

            </goals>

          </execution>

          <execution>

            <id>jacoco-check</id>

            <goals>

              <goal>check</goal>

            </goals>

            <configuration>

              <rules />

            </configuration>

          </execution>

          <execution>

            <!-- Combine the code coverage statistics generated for the unit and the integration tests. -->

            <id>jacoco-merge</id>

            <goals>

              <goal>merge</goal>

            </goals>

            <phase>verify</phase>

            <configuration>

              <fileSets>

                <fileSet>

                  <directory>${project.build.directory}</directory>

                  <includes>

                    <include>\*.exec</include>

                  </includes>

                </fileSet>

              </fileSets>

              <destFile>${project.build.directory}/jacoco-both.exec</destFile>

            </configuration>

          </execution>

          <execution>

            <id>jacoco-integration</id>

            <goals>

              <goal>report-integration</goal>

            </goals>

          </execution>

          <execution>

            <id>jacoco-reportboth</id>

            <goals>

              <goal>report</goal>

            </goals>

            <configuration>

              <dataFile>${project.build.directory}/jacoco-both.exec</dataFile>

              <outputDirectory>${project.reporting.outputDirectory}/jacoco-both</outputDirectory>

            </configuration>

          </execution>

        </executions>

      </plugin>

    </plugins>

  </build>

</project>

Output:

**Mockito Hands-on exercises:**

Exercise 1: Mocking and Stubbing:

ExternalApi.java:

package com.example;

public interface ExternalApi {

    String getData();

}

MyService.java:

package com.example;

public class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

MyServiceTest.java:

package com.example;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

    @Test

    public void testExternalApi() {

        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

}

Pom.xml:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>java-maven-junit-helloworld</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.8.1</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.8.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Needed to run JUnit 5 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

Output:



Exercise 2: Verifying Interactions:

ExternalApi.java:

package com.example;

public interface ExternalApi {

    String getData();

}

MyService.java:

package com.example;

public class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

MyServiceTest.java:

package com.example;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

    @Test

    public void testExternalApi() {

        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        when(mockApi.getData()).thenReturn("Mock Data");

        MyService service = new MyService(mockApi);

        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

}

Pom.xml:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>java-maven-junit-helloworld</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- JUnit 5 -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>5.8.1</version>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.8.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Needed to run JUnit 5 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

Output:

